Parameter Passing

Consider the following program:

```java
public class ParameterDemo {
    public static void main (String[] args) {
        int number = 5;
        System.out.println("main: number is now "+ number);
        int number = 4;
        System.out.println("main: number is now "+ number);
    }
}
```

What's the flow of control?

---

Parameter Passing

Consider the following program:

```java
public class ParameterDemo2 {
    public static void main (String[] args) {
        int number = 5;
        System.out.println("main: number is now "+ number);
        number = 4;
        System.out.println("main: number is now "+ number);
    }
}
```

What's the flow of control?

---

Parameter Passing

Consider the following program:

```java
public class ParameterDemo3 {
    public static void main (String[] args) {
        int number = 5;
        System.out.println("main: number is now "+ number);
        System.out.println("main: number is now "+ number);
    }
}
```

What's the flow of control?

---

Parameter Passing

Consider the following program:

```java
public class ParameterDemo4 {
    public static void main (String[] args) {
        int number = 5;
        System.out.println("main: number is now "+ number);
        System.out.println("main: number is now "+ number);
    }
}
```

What's the flow of control?

---

Parameter Passing

Consider the following program:

```java
public class ParameterDemo5 {
    public static void main (String[] args) {
        int number = 5;
        System.out.println("main: number is now "+ number);
        System.out.println("main: number is now "+ number);
    }
}
```

What's the flow of control?
Parameter Passing

Consider the following program:
public class ParamTest1{
  public static void main (String[] args)
  ...
  x = x * x;
  System.out.println("method1: x is now " + x);  
}

What's the flow of control?

Parameter Passing

Consider the following program:
public class ParamTest2{
  public static void ...
  number = number * number;
  System.out.println("method1: number is " + number);
  }

What's printed?

Parameter Passing

Will this program behave differently?  Why or why not?
public class ParamTest2{
  public static void ...
  number = number * number;
  System.out.println("method1: number is " + number);
  }

What's printed?

Parameter Passing

Consider the following program:
public class ParamTest1{
  public static void main (String[] args)
  ...
  x = x * x;
  System.out.println("method1: x is now " + x);  
}

What's printed?

Parameter Passing

Consider the following program:
public class ParamTest2{
  public static void ...
  number = number * number;
  System.out.println("method1: number is " + number);
  }

What's printed?

Parameter Passing

Consider the following program:
public class ParamTest1{
  public static void main (String[] args)
  ...
  x = x * x;
  System.out.println("method1: x is now " + x);  
}

What's printed?

Parameter Passing

Will this program behave differently?  Why or why not?
public class ParamTest2{
  public static void ...
  number = number * number;
  System.out.println("method1: number is " + number);
  }

What's printed?

Parameter Passing

Consider the following program:
public class ParamTest1{
  public static void main (String[] args)
  ...
  x = x * x;
  System.out.println("method1: x is now " + x);  
}

What's printed?

Parameter Passing

Will this program behave differently?  Why or why not?
public class ParamTest2{
  public static void ...
  number = number * number;
  System.out.println("method1: number is " + number);
  }

What's printed?

Parameter Passing

Consider the following program:
public class ParamTest1{
  public static void main (String[] args)
  ...
  x = x * x;
  System.out.println("method1: x is now " + x);  
}

What's printed?

Parameter Passing

Will this program behave differently?  Why or why not?
public class ParamTest2{
  public static void ...
  number = number * number;
  System.out.println("method1: number is " + number);
  }

What's printed?

Parameter Passing

Consider the following program:
public class ParamTest1{
  public static void main (String[] args)
  ...
  x = x * x;
  System.out.println("method1: x is now " + x);  
}

What's printed?

Parameter Passing

Will this program behave differently?  Why or why not?
public class ParamTest2{
  public static void ...
  number = number * number;
  System.out.println("method1: number is " + number);
  }

What's printed?

Parameter Passing

Consider the following program:
public class ParamTest1{
  public static void main (String[] args)
  ...
  x = x * x;
  System.out.println("method1: x is now " + x);  
}

What's printed?

Parameter Passing

Will this program behave differently?  Why or why not?
public class ParamTest2{
  public static void ...
  number = number * number;
  System.out.println("method1: number is " + number);
  }

What's printed?

Parameter Passing

Consider the following program:
public class ParamTest1{
  public static void main (String[] args)
  ...
  x = x * x;
  System.out.println("method1: x is now " + x);  
}

What's printed?

Parameter Passing

Will this program behave differently?  Why or why not?
public class ParamTest2{
  public static void ...
  number = number * number;
  System.out.println("method1: number is " + number);
  }

What's printed?

Parameter Passing

Consider the following program:
public class ParamTest1{
  public static void main (String[] args)
  ...
  x = x * x;
  System.out.println("method1: x is now " + x);  
}

What's printed?

Parameter Passing

Will this program behave differently?  Why or why not?
public class ParamTest2{
  public static void ...
  number = number * number;
  System.out.println("method1: number is " + number);
  }

What's printed?
Passing object as parameter in Java

- "pass by reference"
  - objects could be huge, so do not pass copies around
  - pass copy of the object reference
  - object reference aka pointer
- modifying object pointed to by reference inside calling method does affect object pointed to by reference outside calling method
  - both references point to same object
Midterm Q4 from 04W2

```java
public void process(int[][] arrA, int[][] arrB) {
    int row;
    int col;
    int[][] arrC = { { 1, 1, 1 }, { 1, 1, 1 } };  
    arrA = arrC;
    for (row = 0; row < arrB.length; row++)
        for (col = 0; col < arrB[row].length; col++)
            arrB[row][col] = row + col;
}
```

dataB
arrA
arrB
1111

Variable Scope

```java
public class CokeMachine4 {  
    private int numberOfCans;  
    public CokeMachine4()  {    
        numberOfCans = 2;   
    }  
}
```

Variable Types

- **Static? Instance? Local? Parameters?**
  - `public class CokeMachine4 {
      private int numberOfCans;
      public CokeMachine4() {
          numberOfCans = 2;
      }
  }`

  - **Static?**
    - `int numberOfCans` is static. It belongs to the class CokeMachine4 and is associated with the class, not an instance.
    - Static fields belong to whole class
    - Static fields can be accessed anywhere in the class.

  - **Instance?**
    - `int numberOfCans` is an instance variable. It belongs to the objects created from the class CokeMachine4.
    - Instance variables can be accessed throughout the object, lifetime of object.

  - **Local?**
    - `int numberOfCans` is local data. It is declared in the `public void process(int[][] arrA, int[][] arrB) {` method.
    - Local variables can be accessed within the method.

  - **Parameters?**
    - `int numberOfCans` is a parameter. It is declared in the `process(int[][] arrA, int[][] arrB) {` method.
      - Parameters can be accessed within the method.
      - Parameters can access static fields associated with the class.

Review: Variable Scope

- **Scope of a variable (or constant) is that part of a program in which value of that variable can be accessed**

Questions?

- `public class CokeMachine4 {
      private int numberOfCans;
      public CokeMachine4() {
          numberOfCans = 2;
      }  
  }`

  - Static variables are declared within the parameter list of the method.
  - Static variables are associated with the class, not an instance.
  - Static variables can be accessed throughout the method, lifetime of the method.

  - Collectors are created to collect data from two separate arrays and store the result in a new array.
  - A for loop is used to iterate through the rows of arrays a and b.
  - Another for loop is used to iterate through the columns of each array.
  - The value of each cell in array c is calculated as the sum of the corresponding cells in arrays a and b.